(12) UK Patent Application (19) GB (11) 2 276 743 (13) A

(43) Date of A Publication 05.10.1994

(21) Application No 9405556.3

(22) Date of Filing 21.03.1994

(30) Priority Data
(31) 05074409 (32) 31.03.1993 (33) JP

(71) Applicant(s)
Fujitsu Limited

(Incorporated in Japan)

1015 Kamikodanaka, Nakahara-ku, Kawasaki-shi, Kanagawa 211, Japan

(72) Inventor(s)
Kenichi Hayashi
Isaac Liu Chuang

(74) Agent and/or Address for Service
Haseltine Lake & Co
Hazlitt House, 28 Southampton Buildings, Chancery Lane, LONDON, WC2A 1AT, United Kingdom

(51) INT CL^5
G06F 15/16 13/38

(52) UK CL (Edition M)
G4A AFGL

(56) Documents Cited
EP 0588104 A2 EP 0544532 A2

(58) Field of Search
UK CL (Edition M) G4A AFGL AMP
INT CL^5 G06F 15/16
ONLINE DATABASE: WPI

(54) Reconfigureable torus network system.

(57) An n-dimensional torus network-based parallel computer in which the torus network is folded n times. Four-terminal switches (30) are placed at folding portions and a switching unit (40) changes over the switches (30) so that any two of the four terminals are linked together. This permits the torus network to be split into subtorus networks or the subtori to be integrated into the original torus network, whereby the reconfiguration of the torus network is realized. The switches are changed over by the switching unit in either a static mode or a dynamic mode.

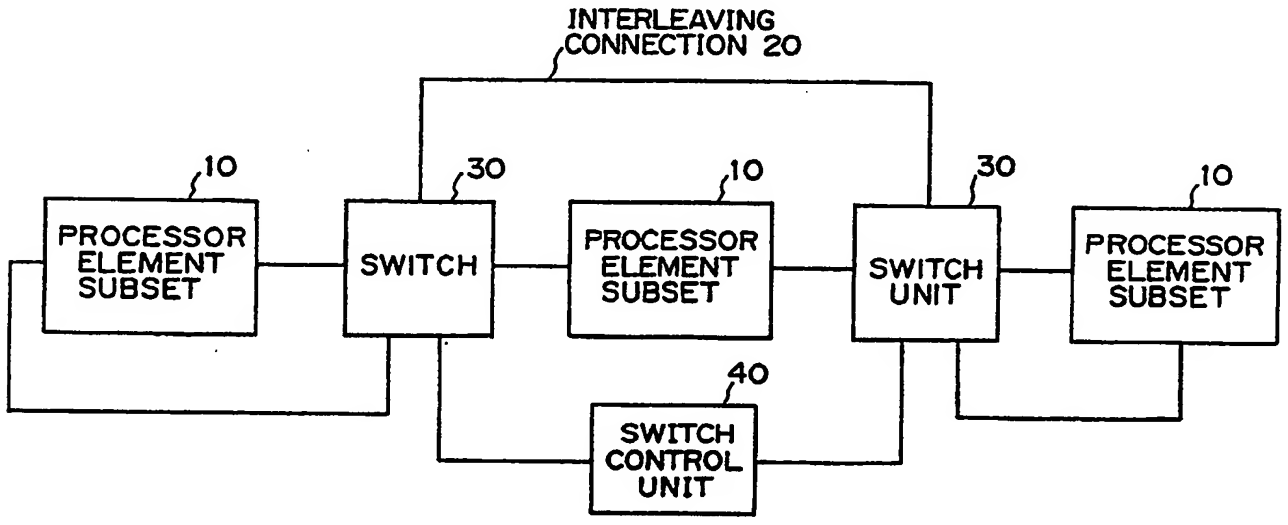


FIG. 2

GB 2 276 743 A

ONE-DIMENSIONAL TORUS NETWORK (PRIOR ART)

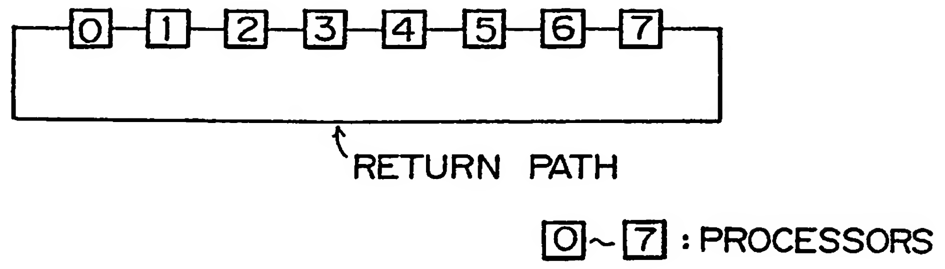


FIG. 1A

SPLITTING METHOD (PRIOR ART)

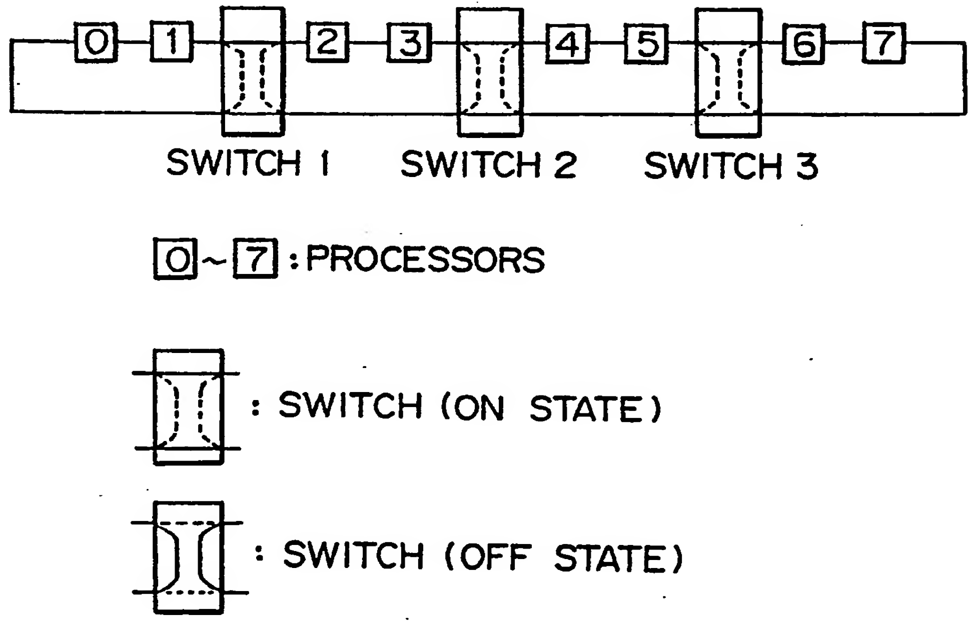


FIG. 1B

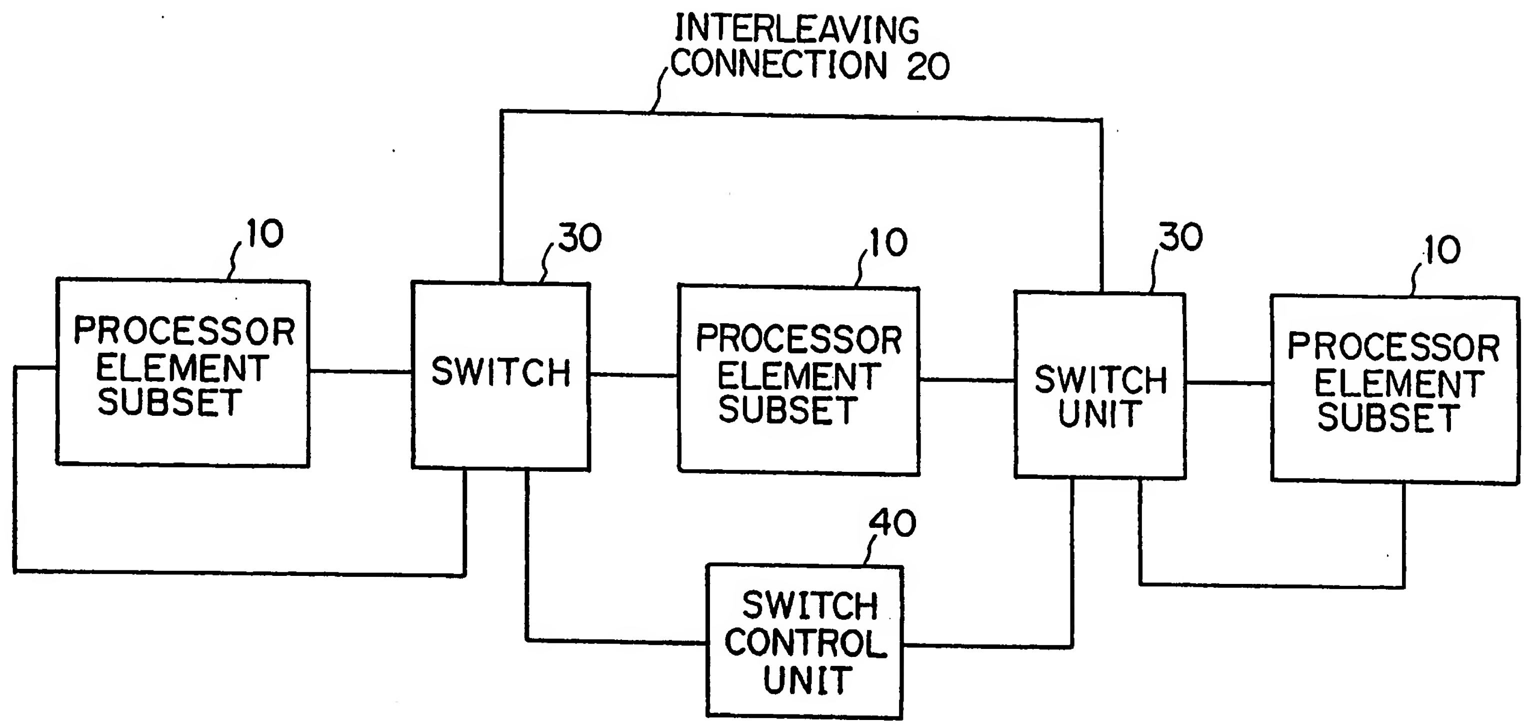

INTERLEAVING CONNECTION 20
10
PROCESSOR ELEMENT SUBSET
30
SWITCH
10
PROCESSOR ELEMENT SUBSET
30
SWITCH UNIT
10
PROCESSOR ELEMENT SUBSET
40
SWITCH CONTROL UNIT

FIG. 2

FIG. 3A

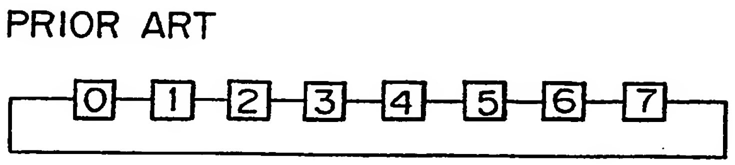


FIG. 3B

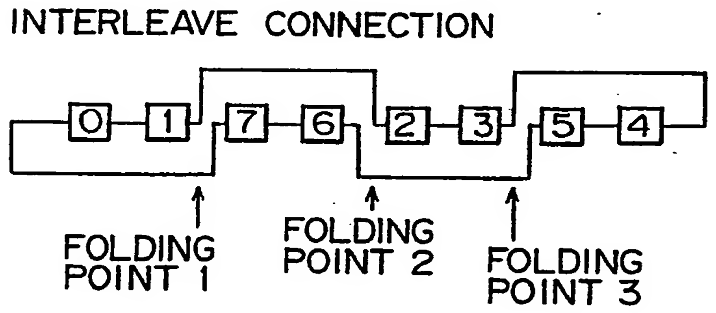


FIG. 3C

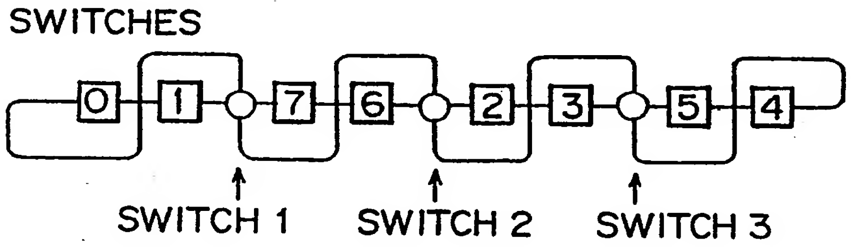
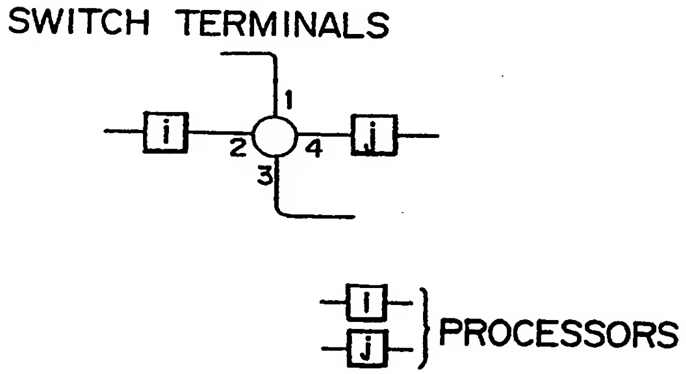


FIG. 3D



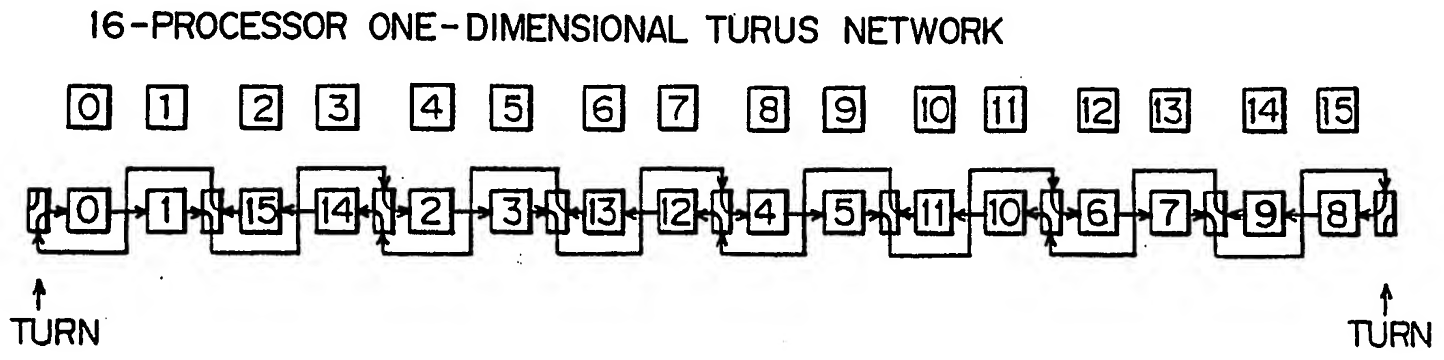

16-PROCESSOR ONE-DIMENSIONAL TURUS NETWORK
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
0
1
15
14
2
3
13
12
4
5
11
10
6
7
9
8
TURN
TURN

FIG. 4A

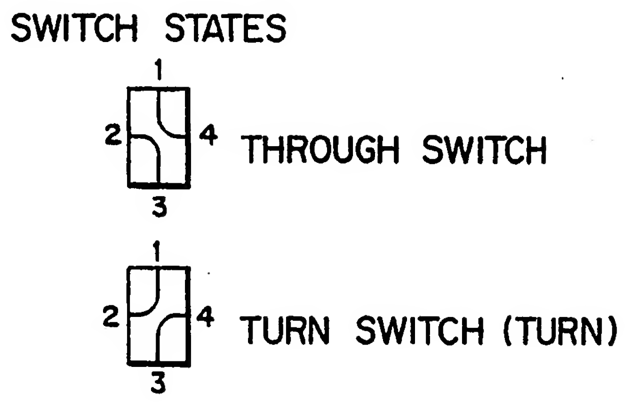

SWITCH STATES
1
2
4
3
THROUGH SWITCH
1
2
4
3
TURN SWITCH (TURN)

FIG. 4B

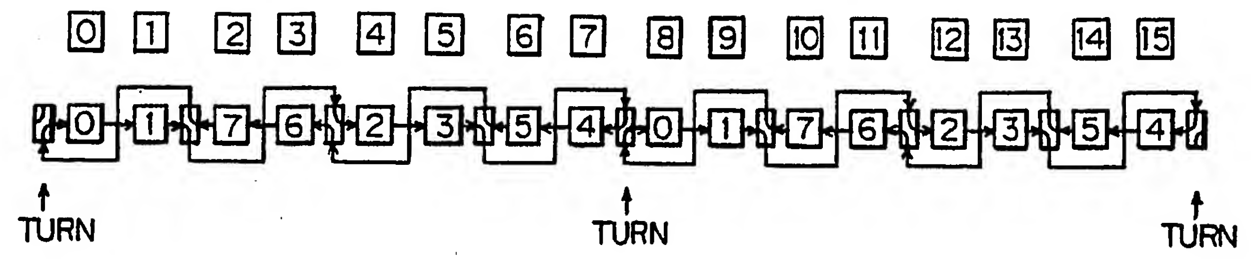


FIG. 5A

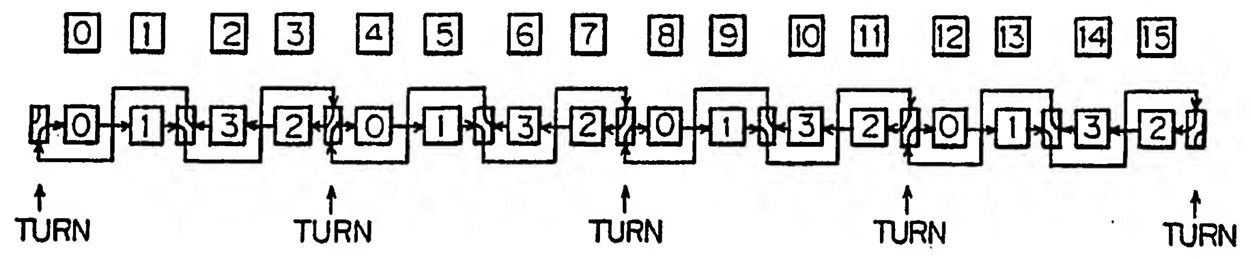


FIG. 5B

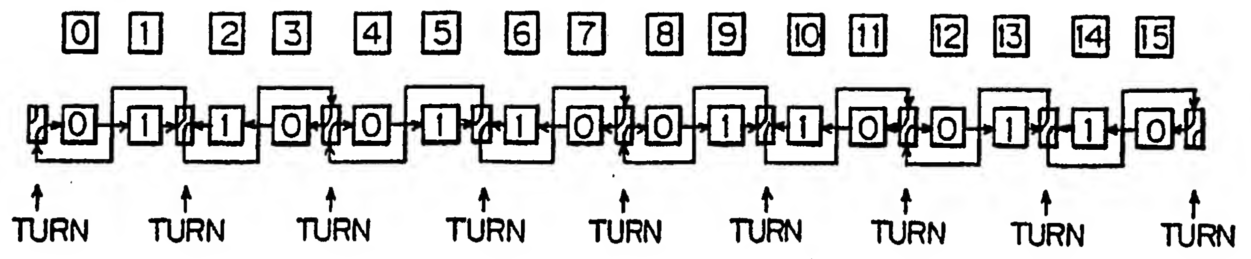
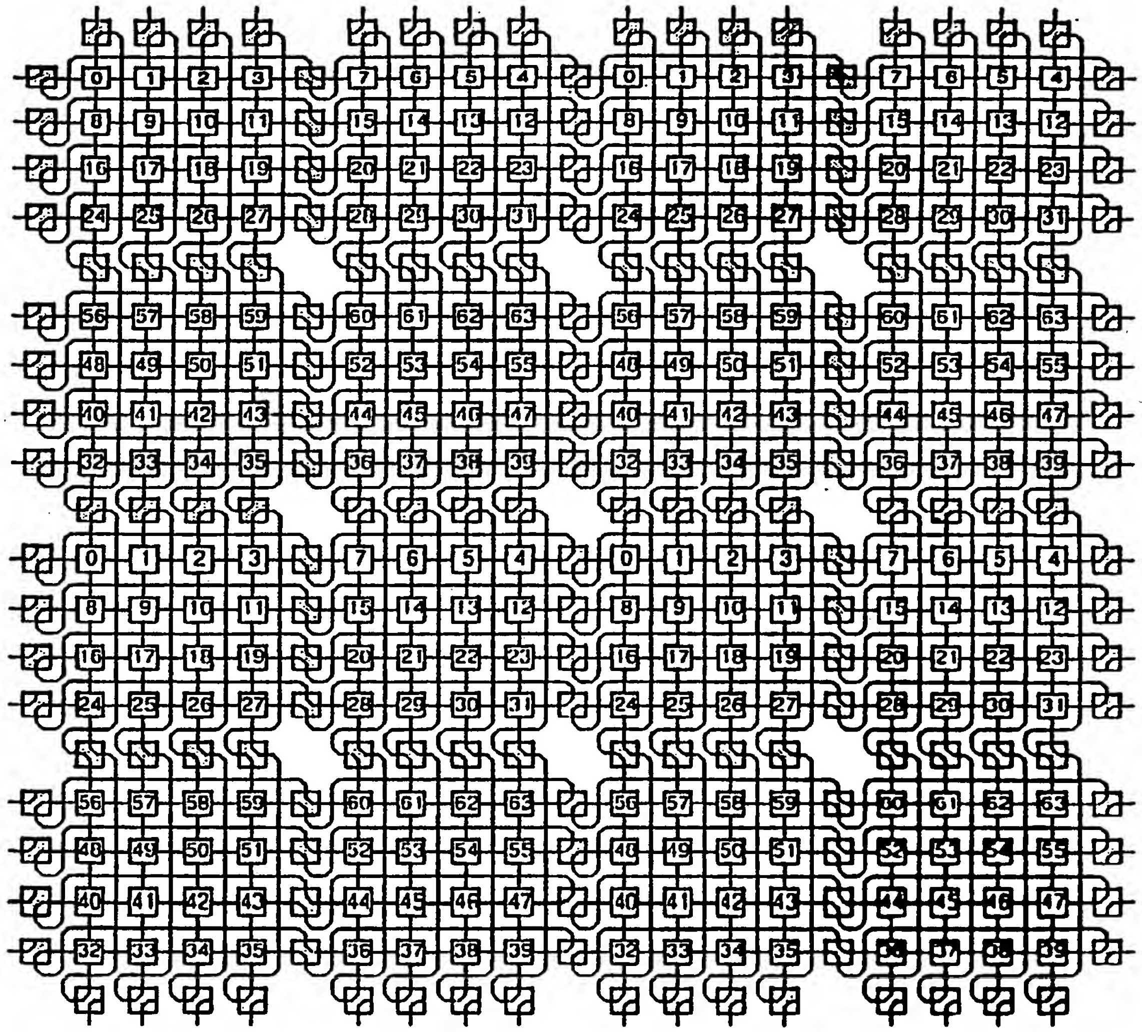


FIG. 5C



$\boxed{0}$ ~ $\boxed{63}$: PROCESSORS (NUMBERS REPRESENT LOGICAL ID)

: THROUGH SWITCH

: TURN SWITCH

FIG. 6

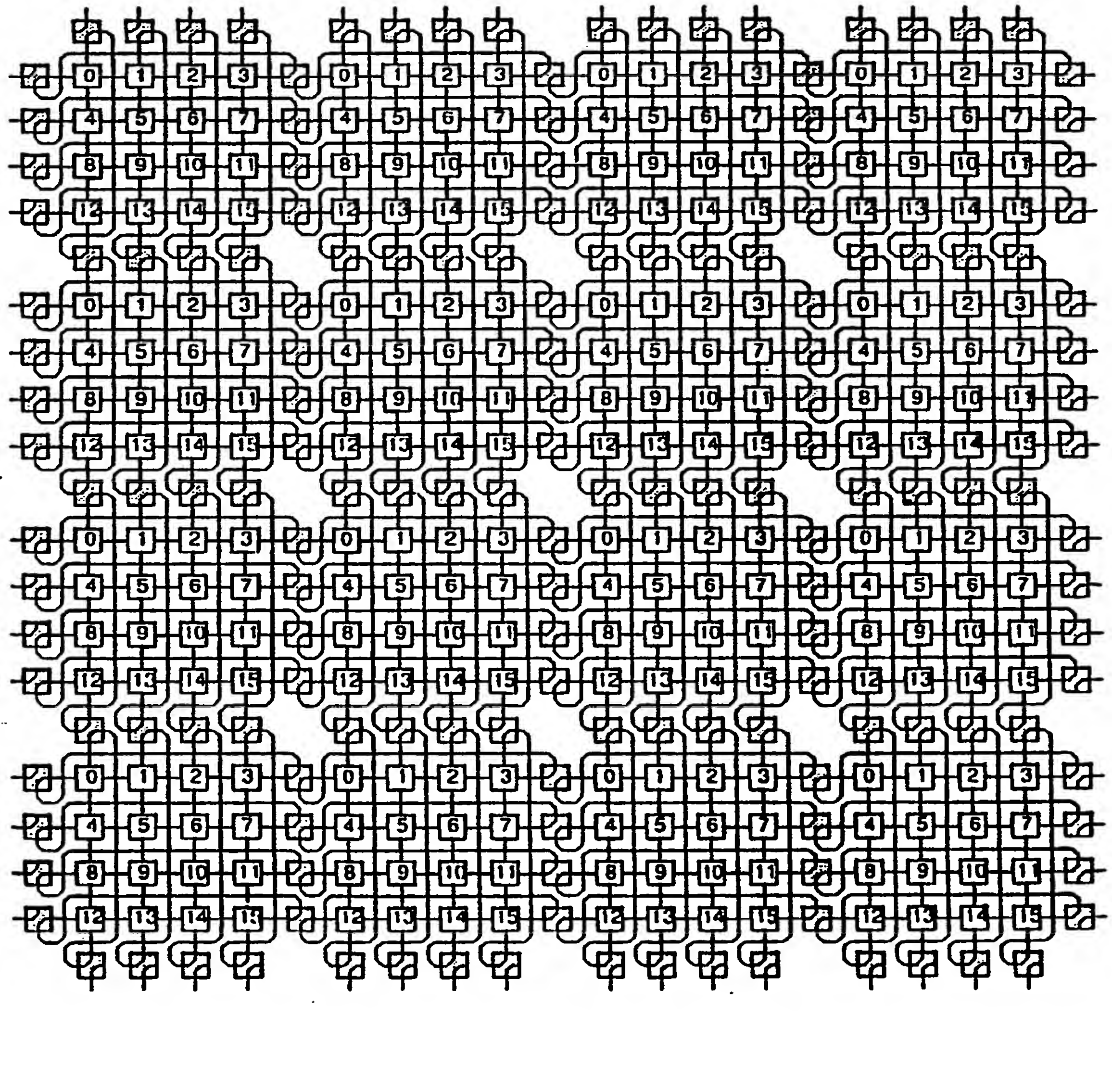


FIG. 7

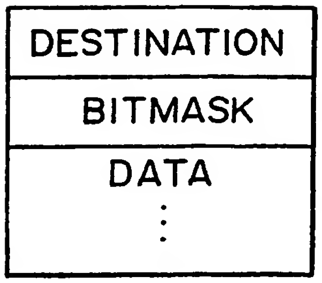


FIG. 8A

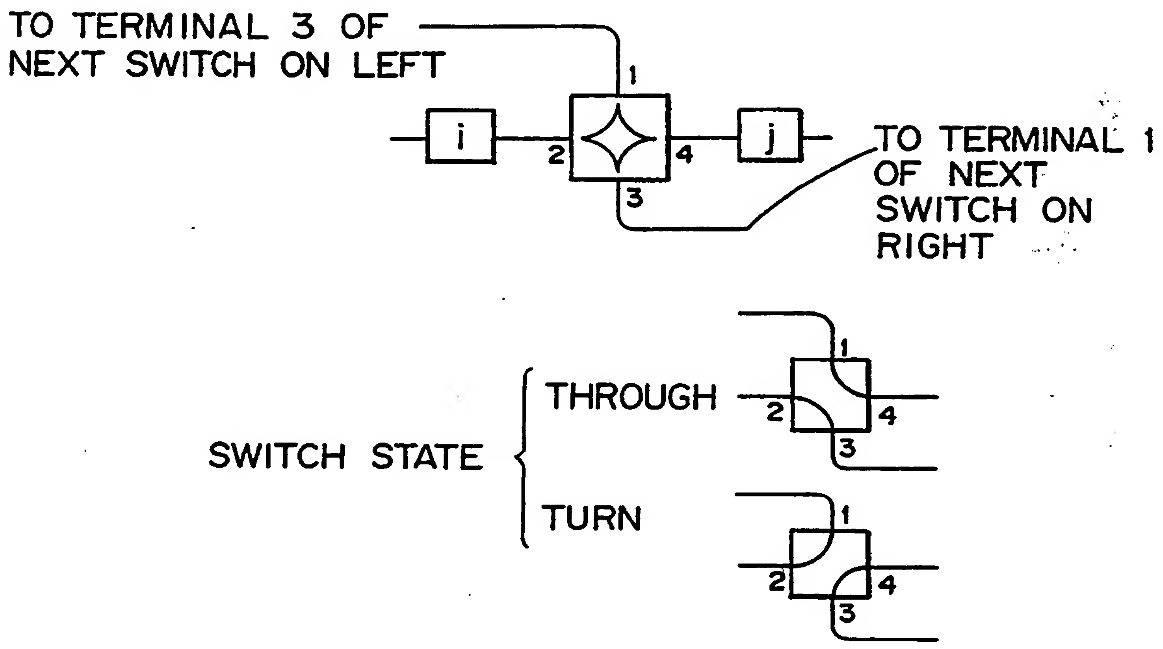

TO TERMINAL 3 OF NEXT SWITCH ON LEFT
1
i
2
4
j
3
TO TERMINAL 1 OF NEXT SWITCH ON RIGHT
SWITCH STATE
THROUGH
1
2
4
3
TURN
1
2
4
3

FIG. 8B

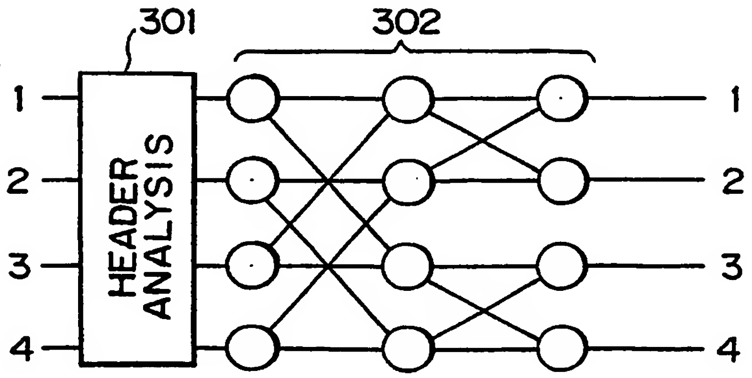
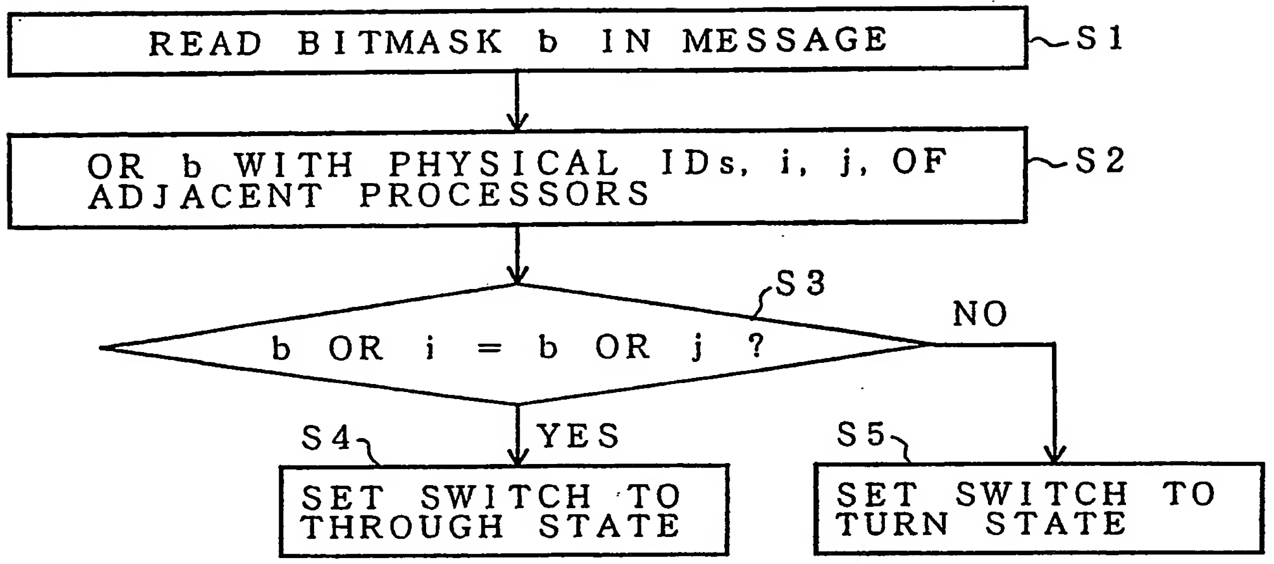
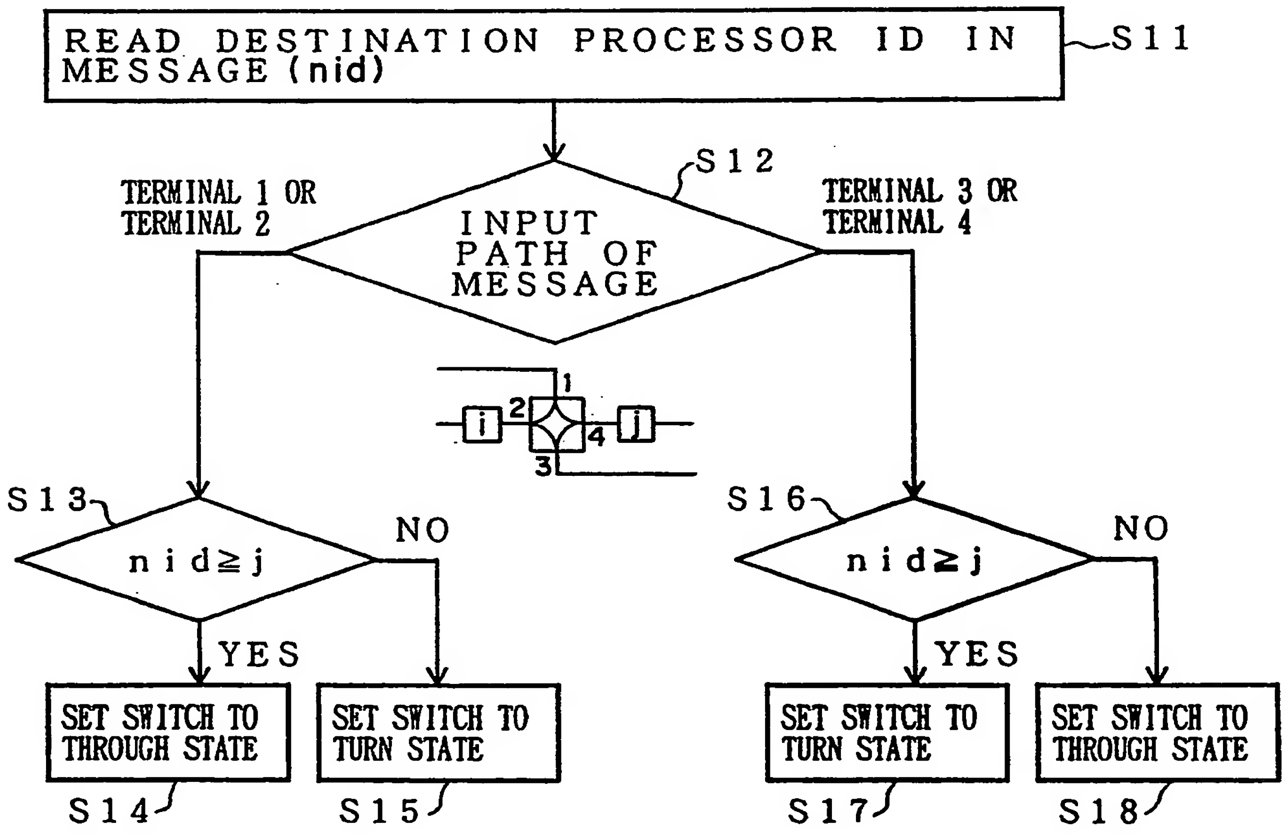


FIG. 8C



READ BITMASK b IN MESSAGE
S1
OR b WITH PHYSICAL IDs, i, j, OF ADJACENT PROCESSORS
S2
S3
b OR i = b OR j ?
NO
YES
S4
SET SWITCH TO THROUGH STATE
S5
SET SWITCH TO TURN STATE

FIG. 9A



READ DESTINATION PROCESSOR ID IN MESSAGE (nid)
S11
S12
TERMINAL 1 OR TERMINAL 2
INPUT PATH OF MESSAGE
TERMINAL 3 OR TERMINAL 4
1
2
i
4
j
3
S13
nid≧j
NO
YES
SET SWITCH TO THROUGH STATE
S14
SET SWITCH TO TURN STATE
S15
S16
nid≧j
NO
YES
SET SWITCH TO TURN STATE
S17
SET SWITCH TO THROUGH STATE
S18

FIG. 9B

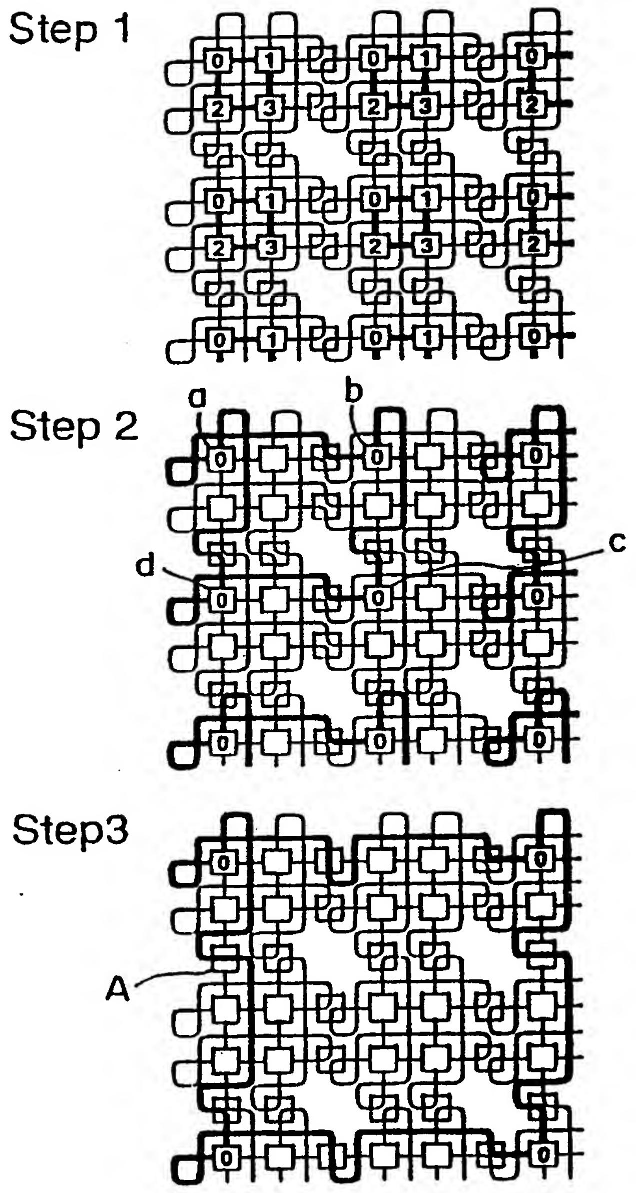
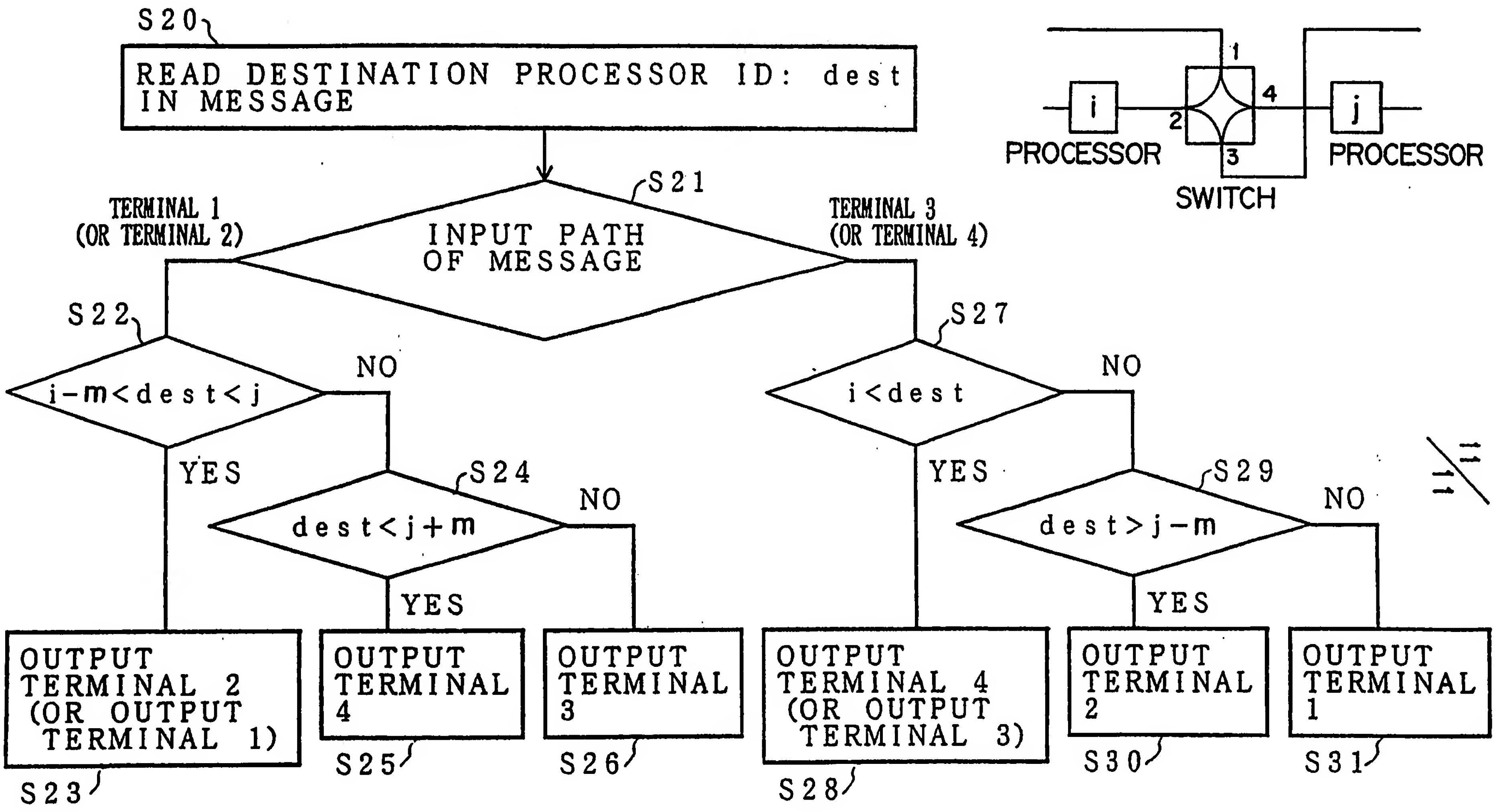


FIG. 10



NOTE: "m" REPRESENTS THE SIZE OF THE SUBTORUS IN THE SMALLEST UNIT: m = 2 WHEN THE SMALLEST UNIT COMPRISES 2 x 2 PROCESSORS.

FIG. 11

Reconfigurable Torus Network System

Background of the Invention

Field of the Invention

The present invention relates to a reconfigurable torus network system used in a parallel computer and, more particularly, to a reconfigurable torus network system which is capable of being split into a plurality of subsystems having subtori of the same phase for availability to a plurality of users, and high-speed global calculations and broadcast processing.

Description of the Related Art

Coupling techniques for parallel computers include mesh, binary n-cube, hyper-cube and torus networks. When the cost of hardware is assumed to be constant, of k-ary n-cube networks, low-dimensional torus networks of, say, two or three dimensions are known as good network topology for achieving a low level of latency and high throughput.

Figures 1A and 1B are schematic diagram of a prior art torus network. More particularly, Figure 1A shows an example of a one-dimensional torus network in which eight nodes are interconnected in a torus network.

A feature of the torus network is that the end processors (processor 0 and processor 7 in Figure 1A) are connected together. The path connecting the end processors is called a return path. When such a torus network is implemented as hardware, a problem arises in that the wire length of the return path has to become very long because the return path wire wraps around.

In a parallel computer, it is desirable that a number of processors constituting the parallel computer be split into several processor groups for availability to a plurality of users. When a mesh network-based parallel computer is split to make it available as a plurality of systems, processors are reconfigured to form networks of the same phase, thereby allocating one system to each subnetwork.

As a method of splitting a torus network, a recursive torus coupling architecture has been proposed so far (Matsuyama, Aoyama: Recursive Torus Coupling Architecture, The Technical Research Report of the Institute of Electronics Information and Communication Engineers Computer System, CPSY91-4-33, 1991, pp. 49-58).

In this system, a recursive hierarchical structure is embedded in a torus network by placing on/off switches on the network. Figure 1B is a conceptual diagram illustrating this system in terms of a one-dimensional torus network comprised of eight processors.

In this figure, three switches are placed such that the eight processors can be split into a maximum of four groups. Each of the switches is placed such that it spans the return path, which links processor 7 and processor 0 together, between two processors (as an example, processor 1 and processor 2). When a switch is in the OFF state, direct connection is established between the corresponding processor and the return path, thereby splitting the network. When the switch is in the ON state, on the other hand, direct connection is made between the two processors and the return path is recovered to thereby restore the original network. For example, when switches 1, 2 and 3 are all placed in the OFF state, the network is split into four sections. When these switches are all placed in the ON state, on the other hand, a single network is restructured.

However, this system will not solve the problem with the torus network that the return path is long.

When an n-dimensional torus network is simply implemented, the return paths connecting ends of the n-dimensional mesh wrap around. In the case of the above split system, long return paths are left as they are, and switches are inserted in each return path. Each switch is required to be connected with the path between processors as well as a return path (Figure 1B).

As described above, the prior art split method suffers from problems that long return paths exist and switches have to be inserted at points on the return paths.

Summary of the Invention

It is therefore an object of the present invention to provide a torus network, in which the ends of a mesh are connected together, which permits subnetworks of the same phase to be reconfigured with a structure in which long return paths are distributed, makes a parallel computer available to a plurality of users, and permits global calculations and broadcast processing to be performed at high speed.

The present invention supposes an n-dimensional torus network-based parallel computer in which processor elements are interconnected in an n-dimensional torus network. Further, the present invention supposes the provision of switches for changing over the network.

The torus network comprises a plurality of processor element subsets, each comprising one or more processor elements. An interleaving connecting unit is provided which interleaves processor-elements and embeds the results of an interleave connection in one n-dimensional layer. Thereby, the processor elements are interconnected by the interleave connection, processor elements at both ends in a usual torus network are placed close to each other and, the length of the return path is distributed throughout the network, thus reducing the length of the return path connecting the processor elements at both ends in a usual torus network.

An ON/OFF switch units are placed at interleaving connecting points of the torus network which is interleave-connected by the interleaving connection unit. Each of the switch units are linked to two processor elements and two adjacent switches on the interleaved torus network. That is, each switch unit has four terminals. The two terminals of them are used as input terminals and the two other terminals are used as output terminals.

A switch control unit is provided which changes over the switch units. This switch control unit changes over the switch units so that any one of the input terminals is connected to either of the output terminals. This permits the torus network to be split into processor elements substes or subtorus networks or permits the processor element substes or the subtorus networks to be integrated into or the original torus network.

According to the present invention, the interleave connecting unit permits a long return path inherent in a usual torus network to be eliminated, an irregular thereby enabling delay in communications through the long return push to be decreased and the switch unit permits the split/integration (reconfiguration) of the torus network. By dynamically splitting/integrating the torus network, global calculations and broadcast processing can be speeded up.

Brief Description of the Drawings

Other objects and advantages of the present invention will be apparent from the following description of preferred embodiments of the invention in conjunction with the accompanying drawings in which:

Figures 1A and 1B illustrate prior art method;

Figure 2 is a basic block diagram of the present invention;

Figures 3A to 3D are diagrams for use in explanation of the principle of the present invention;

Figures 4A and 4B illustrate system configurations of a preferred embodiment of the present invention;

Figures 5A to 5C illustrate examples of ways in which a torus network is split;

Figure 6 illustrates a two-dimensional

reconfigurable torus network (a reconfigurable torus network split into four 8 x 8 subunits);

Figure 7 illustrates a two-dimensional reconfigurable torus network (a reconfigurable torus network split into sixteen 4 x 4 subunits);

Figures 8A to 8C are diagram for use in explanation of a switch control system in a dynamic mode;

Figures 9A and 9B are flowcharts illustrating the operation of a header analyzing unit in the dynamic mode;

Figure 10 is a diagram for use in explanation of global calculations; and

Figure 11 is a flowchart illustrating the operation of the header analyzing unit in an express torus.

Detailed Description of the Preferred Embodiments

Basic Configuration

Figure 2 is a basic block diagram for explaining the principle of the present invention.

The present invention supposes an n-dimensional-torus-network-based parallel computer. Further, the parallel computer supposes the provision of switches for switching the torus network.

The torus network comprises a plurality of processor element substes. An interleaving connection unit 20 interleaves processor element subset and embeds the results of the interleave connection in one dimensional layer for connection. Processor elements are interconnected in an interleaved torus network.

Switch units 30 switch on or off connections at respective interleaving connection points in communication paths of the torus network which is connected by the interleave connection unit 20.

A switch control unit 40 changes over the connections of the switch units 30, as required.

Next, the operation of the functional block diagram shown in Figure 2 will be described.

Figures 3A to 3D illustrate the principle of the present invention. For the purpose of simplifying the description, consider a one-dimensional torus network that is composed of eight processors.

For comparison there is illustrated a conventional torus network in Figure 3A. The eight processors numbered 0 through 7 are connected in cascade, and the processors numbered 0 and 7 are linked together by a return path, whereby a doughnutlike torus network is formed.

The conventional torus network shown in Figure 3A is formed into an interleaving or splittable network configuration by the interleave connecting unit 20 and the switch unit 30 as shown in Figures 3B and 3C.

That is to say, the conventional torus network is interleaved by the interleaving connection unit 20 (see Figure 3B). In this case, the network is interleaved such that the processors are handled in pairs or sets in order to make the smallest interleave or split configuration of a processor element subset comprise two processors. The interleaving connection starts or ends at points 1, 2 or 3. It should be noted here that the minimum interleaving or split configuration need not be restricted to two processors.

By the switch unit 30 switches 1, 2, and 3 are inserted into the points 1, 2, and 3 as shown in Figure 3C. In this exemplary system in which the minimum processor configuration of each processor element subset or subtorus comprises two processors, switches 1, 2, and 3 are respectively placed between processors 1 and 7, between processors 6 and 2, and between processors 3 and 5. If the smallest processor configuration of each processor element subset or subtorus comprises four processors, then the interleaving connection will start or ends at one point, in which case a switch is placed only in the position of switch 2 shown in Figure 3C.

The switch control unit 40 changes over the switch units 30 in the interleaving or splittable torus network thus formed. As shown in Figure 3D, each switch unit 30 has four terminals 1, 2, 3, and 4 which are respectively connected to two processors and two adjoining switch units. When, in Figure 3C, switches 1, 2 and 3 are changed over so that connections are made between terminal 1 and terminal 2 and between terminal 3 and terminal 4, the torus network composed of eight processors can be split into four processor element subsets. The switch unit 30 may be composed or a known crossbar switch or butterfly switch.

When switches 1 and 3 are changed over so that connections are made between terminal 1 and terminal 4 and between terminal 2 and terminal 3, and switch 2 is changed over so that connections are made between terminal 1 and terminal 2 and between terminal 3 and terminal 4, the torus network comprised of eight processors can be split into two processor element subsets. Therefore, as is clear from Figure 3C, the long return push is eliminated from the torus network.

The changing over of the switches may be performed at regular time intervals by the switch control unit 40 provided in system management side of the parallel computer in a static mode in which the configuration of subtori is changed on a time-sharing basis to allow several users to use the subtori on a time-sharing basis. The changing over of the switches may be performed dynamically by the use of messages issued by users in a dynamic mode.

For communications between subtori a path connecting switches can be used, which permits faster communications than the conventional torus network (i.e., an express torus can be implemented). By this express torus, global calculations and broadcast processing, which are performed frequently in a parallel computer, can be performed at high speed.

Description of the Preferred Embodiments

Hereinafter, the preferred embodiments of the present invention will be described in conjunction with the drawings.

The torus-network splitting method of the present invention can be applied to multi-dimensional torus networks as well as one-dimensional torus networks. Also, the present invention can be applied not only to square networks but also to rectangular networks.

Here, the splitting of a one-dimensional torus network will be described first, and an extension of the splitting method to a two-dimensional network will be described next.

Figures 4A and 4B illustrate system configuration of a one-dimensional torus network.

In this example, 16 processors numbered 0 through 15 are interconnected in a one-dimensional torus network. The network is interleaved such that the smallest unit of split, i.e. the smallest processor element subset comprises two processors. In the figure, the upper numbers represent physical processor identification (ID) numbers, while the numbers within the network represent logical processor identification numbers. By interleaving the network, the processors are logically sequenced such that 0 - 1 - 15 - 14 - 2 - 3 - 13 - 12 - 4 - 5 - 11 - 10 - 6 - 7 - 9 - 8. By interleave-connecting these processors and placing switches as shown in Figure 4A, along return path of the torus network is avoided and a reconfigurable torus network is formed.

Since the smallest reconfiguration unit comprises two processors, a switch is inserted every two processors. Each switch has two possible states in order to reconfigure the torus network. Figure 4B shows the two possible states of the switch (THROUGH state and TURN state). The numbers (1 to 4) in the switch symbols indicate input and output terminals of the switch. The THROUGH state is the switch state that permits the processor element subset to be interleaved and also permits the subtori to be locally integrated into the original torus network. The TURN state is the switch state that permits the torus network to be split logically into subtori.

In Figure 4A, the switches embedded every two processors are all set to the THROUGH state. For example, the THROUGH switch between processor 3 and processor 13 (represented by logical processor ID here) has its terminal 2 connected to processor 3, its terminal 4 connected to processor 13, its terminal 1 to the next THROUGH switch on the left side, and its terminal 3 connected to the THROUGH switch on the right side, thus linking two subtori respectively comprising processors 2, 3 and processors 13, 12.

TURN switches are placed at the opposite ends of the network. The left TURN switch connects terminal 1 of the next THROUGH switch on the right side and processor 0 together, thus terminating the left end of the torus network. Similarly, the right TURN switch connects terminal 3 of the next THROUGH switch on the left side and processor 8 together, thus terminating the right end of the torus network.

As can be seen from the foregoing, by changing over each switch placed between processors between the two states of THROUGH and TURN, subtori can be combined into a torus network or a torus network can be split into subtori, thereby permitting the restructure of the torus network.

Figures 5A to 5C illustrate examples of reconfiguration of the one-dimensional 16-processor torus network of Figure 4A which are obtained by changing over the switches.

In Figure 5A, only the switch placed between processor 7 and processor 8 (hereinafter represented by physical processor IDs) is changed over to the TURN state. As a consequence, the 16-processor torus network is split into two 8-processor subtori. If, in this state, the switch between processor 7 and processor 8 is changed over to the THROUGH state, then the torus network will be reconfigured into the one-dimensional 16-processor torus network.

Similarly, when the respective switches between processor 3 and processor 4, between processor 7 and processor 8 and between processor 11 and processor 12 are changed over to the TURN switch state, the torus network is split into four 4-processor subtori as shown in Figure 5B.

Further, if all the switches of the reconfigurable torus network are changed over to the TURN switch state, then the torus network will be split into eight 2-processor subtori as shown in Figure 5C.

As described above, by interleave-connecting a processor element subset through the THROUGH state of the switch, the long return push of the torus network is eliminated and by changing over the switches between the TURN state and the THROUGH state, the torus network can be split into some subtorus network or some subtorus network can be combined into one torus network.

Figs. 6 and 7 illustrate embodiments of two-dimensional reconfigurable torus networks, both of which comprise 16 x 16 processors.

In Figure 6, the smallest split unit comprises 4 x 4 processors and switches are placed around them. In this case, the switch states are determined such that the network is split into four 8 x 8 subtori (the processor logical ID numbers in each subtorus are 0 through 63). That is to say, the switches inside each subtorus are set to the THROUGH state, while the switches around the subtorus, i.e., the switches between adjacent subtori are set to the TURN state.

In the network of Figure 7, as in the network of Figure 6, the smallest split unit comprises 4 x 4 processors. In this network, the switch states are determined such that the network is split into sixteen 4 x 4 processor subtorus (the processor logical ID numbers in each subtorus are 0 through 15). That is to say, all the switches in the network are placed in the TURN state.

Next, a method of determining the switch states explained so far, i.e., a method of restructuring a reconfigurable network will be described.

A reconfigurable torus network can be restructured by the two following methods: a pseudo-static mode and a dynamic mode.

In the pseudo-static mode, the system side, for example, a system manager controls the state of each switch. This mode is a simple yet adequate method to realize the multi-user environment of a parallel computer.

The time is broken into time blocks of, say, several milliseconds, and the switch state is changed every time block to change the configuration of subtori. Each of the time blocks is allocated to a separate user, thereby realizing the multi-user environment. For example, in a distributed-memory parallel computer (DMPP) with 64 x 64 processors, a user is allowed to use the whole DMPP as the 64 x 64 torus at a certain phase, and four different users are allowed to use each of four 16 x 16 subtori at an another phase.

In the pseudo-static mode described above, the system simply sets the state of each switch to the TURN or THROUGH state prior to each phase. Within one phase, i.e. within a certain time period the switches are each placed in the fixed state. The routes via the switches in the processor-to-processor communications paths are fixed within that phase.

In the dynamic mode, on the other hand, the state of each switch is changed by a message communicated by a user among processors to reconfigure the torus network. Upon receipt of that message each switch analyzes the contents of the message and then changes its state accordingly, thereby restructuring the torus network.

Figures 8A to 8C illustrate the switch controlling method in the dynamic mode.

Figure 8A shows the structure of a message transmitted by a user, which consists of a header part and a data part. The header part comprises a physical processor ID (Destination) indicating the destination of the message and numeric data (Bitmask) representing the subtorus's size. The numeric data Bitmask has numeric values corresponding in number to the dimension of the subtorus. For example, assume that a one-dimensional subtorus has m processors. Then, its Bitmask value b will be b = m - 1. In the case of an n-dimensional subtorus, n Bitmask values bk (k = 1, 2, ,n) are provided. With a two-dimensional subtorus having 4 x 4 processors, Bitmask values b1 and b2 are b1 = 3 and b2 = 6.

Upon receipt of a message of Figure 8A transmitted by a user, a switch analyzes the header information and then determines which of the TURN state and the THROUGH state it is to be set to.

Figure 8B illustrates the symbol of a switch. The switch is interposed between processor i and processor j. Processor i is connected to terminal 2 of the switch, while processor j is connected to terminal 4 of the switch. Switch terminal 1 is connected to terminal 3 of the next switch on the left side, while switch terminal 3 is connected to terminal 1 of the next switch on the right side.

When the switch is to be set to the THROUGH state as a result of the analysis of the header information, connections are made between terminal 1 and terminal 4 and between terminal 2 and terminal 3. When the switch is to be set to the TURN state, on the other hand, connections are made between terminal 1 and terminal 2 and between terminal 3 and terminal 4.

Figure 8C shows an embodiment of the switch. The switch can be implemented by a header analyzing section 301 and a butterfly network 302, for example. When the result of analysis by the header analyzing section 301 indicates THROUGH, the butterfly network 302 is controlled so that inputs 1, 2, 3, 4 are coupled to outputs 4, 3, 2, 1, respectively. If, on the other hand, the result of analysis made by the header analyzing section 301 indicates TURN, then the butterfly network 302 is controlled so that inputs 1, 2, 3, 4 are coupled to outputs 2, 1, 4, 3, respectively.

Figures 9A and 9B are flowcharts illustrating the operation of the header analyzing section.

The flowchart of Figure 9A is directed to a method of controlling the switch using Bitmask values in the header of a message.

When a message transmitted by a user arrives at the switch, the header analyzing section 301 reads a Bitmask value b in the header of the message (step S1). The value b is ORed with each of i and j, which represent the physical IDs of the adjacent processors, bit by bit (step S2). If, for example, the size of a one-dimensional subtorus is eight processors, and the physical IDs of the adjacent processors are 3 and 4, then b = 7 ("0111" in binary), i = 3 ("0011" in binary), and j = 4 ("0100" in binary). Hence b OR i = 7 ("0111" in binary) and b OR j = 7 ("0111" in binary). If, for example, the size of a one-dimensional subtorus is eight processors, and the physical IDs of the adjacent processors are 7 and 8, then b = 7 ("0111" in binary), i = 7 ("0111" in binary), and j = 8 ("1111" in binary). Hence b OR i = 7 ("0111" in binary) and b OR j = 8 ("1111" in binary). In this case, an imaginary processor is considered next to the end of the one-dimensional subtorus and the physical ID of the imaginary processor is j which is equal to i + 1, i.e. 8.

Next, a decision is made as to whether or not the b OR i value and the b OR j value are equal to each other (step S3). If they are equal (YES), then the switch is set to the THROUGH state (step S4). On the other hand, if they are not equal (NO), then the switch is set to the TURN state (step S5).

According to this control method, the user is allowed to restructure the subtorus by specifying the Bitmask value. For example, in the case of the one-dimensional torus network of Figure 4A in which 16 processors are included and the smallest subtorus configuration comprises two processors, if the user sets Bitmask value b to b = 7, then the results will be

b OR i ≠ b OR j for the switch between processor 7 and processor 8

and b OR i = b OR j for other switches.

As a result, the switch between processor 7 and processor 8 is set to the TURN state, while the other switches are set to the THROUGH state.

If the user sets so that Bitmask value b = 3, then the results will be

b OR i ≠ b OR j for the switches between processor 3 and processor 4, between processor 7 and processor 8, and between processor 11 and processor 12 and b OR i = b OR j for other switches.

As a result, the switches are controlled so that the switches between processor 3 and processor 4, between processor 7 and processor 8 and between processor 11 and processor 12 are set to the TURN state, and the other switches are set to the THROUGH state.

Figure 9B is a flowchart illustrating the operation of a control method using a Destination value in the header of a message.

When a message transmitted by a user arrives at a switch, the header analyzing section reads a Destination value n id in the header of the message (step S11). The n id represents the physical ID of a destination processor.

Next, a decision regarding message input path is made (step S12). The subsequent processing varies according to the input path of that message. When the message is routed via terminal 1 or terminal 2 (processor i) of the switch, a comparison is then made between the n id value and the processor physical ID j (S13). If n id ≥ j (YES), then that switch is controlled so that it is set to the THROUGH state (S14). Conversely, if n id < j (NO), then that switch is controlled so that it is set to the TURN state (step S15).

When the message is routed via terminal 3 or terminal 4 (processor j) of the switch, on the other hand, a comparison is made likewise between the n id value and the processor ID value j (step S16). If n id ≥ j (YES), then that switch is controlled so that it is set to the TURN state (S17). Conversely, if n id < j (NO), then that switch is controlled so that it is set to the THROUGH state (step S18).

As described above, by controlling switches using a destination processor ID (n id) contained in a message, the state of each of switches placed between a processor from which that message is transmitted by a user and the destination processor can be changed over.

Next, a method of performing global calculations and broadcast processing at high speed by using a control method that is an extension to the dynamic mode switch control method for the global calculations and the broadcast processing.

Referring back to the network of Figure 6 in which 16 x 16 = 256 processors are divided into four subunits by switches, processor 0 and processor 7, in the 8 x 8 subunit at the upper left, are linked together via a turn switch connected to processor 0 and a through switch connected to processor 7 with a distance of 1 therebetween. Here, the distance between adjacent processors is assumed to be 1. In the routing in a normal torus network, the distance between processor 0 and processor 7 is 4 because processors 1, 2 and 3 are interposed therebetween. It will therefore be appreciated that a route via the above-recited turn and through switches is a shortcut.

Communications delays include delays caused by wires linking processors and delays originating in processors. The processor delays usually are considerably larger than the wire delays. If, therefore, use is made of a communications path that is provided with a smaller number of nodes in the middle, then it will be possible to decrease delays. In the present system, any path via switches is linked to a remote processor without passing through intermediate processors, which permits a message to be routed at high speed to any remote processor. Such a path is here called an express torus.

The use of this express torus permits global calculations and broadcast processing, which usually require a large amount of time for communications because of long distance, to be completed in a short period.

The global calculations, which obtain the sum of values of certain variables, maximum values, minimum values, etc. in all processors, are a commonly used calculation pattern in a parallel computer.

The basic algorithm of global calculations using an express torus is hierarchical.

Figure 10 is a diagram explanatory of global calculations using an express torus.

A reconfigurable torus network in the figure is of two-dimensions the smallest unit of which comprises 2 x 2 processors. Although 5 x 5 processors are shown in the figure, they form only a part of that torus network. In fact, such network parts are extended.

First, global calculations are performed by the smallest unit of network split comprised of 2 x 2 processors, and a representative processor among them, for example, processor 0, waits for the results of calculations (Step 1). Next, the representative processors in the four units (processors 0 in the four units, indicated at a, b, c, d in Step 2) interchange the results of global calculations and then perform further global calculations the results of which are held by a representative processor (for example, processor a) (Step 2). At this point, data interchange among the processors is performed through the express torus via switches. As can be seen from the figure, each of the processors a, b, c and d is adjacent to another with a switch interposed therebetween.

As the above processing is repeated, the unit area where the global calculations are performed extends gradually to a larger subunit (5 x 5 processors in Step 3). Finally, when the results of the whole global calculations are obtained, the results are sent from an upper hierarchical level to a lower hierarchical level, i.e., in the direction opposite to that on calculations. This permits all the processors to hold the results of the whole global calculations.

Like the global calculations, the broadcast processing can also be performed using an express torus. That is, a broadcast request is first issued to a processor at the highest level in the hierarchy. Then, data is sent from an upper hierarchical level to a lower hierarchical level in sequence. This permits the data to be sent to all of the processors.

In order to use an express torus for global calculations and broadcast processing, it is necessary to extend the switch control system in the dynamic mode. That is, although the switch state in the dynamic mode is restricted to the THROUGH and TURN states, it is required that, in the express torus, any two of the four terminals of each switch be capable of being linked together. In the switch indicated at A in the Step 3 of Figure 10, terminal 1 and terminal 3 are shown linked together. An express torus cannot be implemented without being capable of linking any two of the switch terminals together in such a manner. Explaining the extended switch control system in terms of the embodiment of the switch shown in Figure 8C, the switch is controlled so that input terminal 1, for example, can be linked to any one of output terminals 2, 3 and 4.

Figure 11 is a flowchart for the header analyzing operation in an express torus in the network shown in Figure 10.

When a message transmitted from a processor arrives at a switch, information on the destination of that message (physical processor ID:dest) in the header part is read first (step S20).

Next, a message input path decision is made (step S21). The subsequent processing varies according to the input path of the message. When the message is routed via terminal 1 or terminal 2 (processor i), a comparison is made among the destination processor ID (dest), i - m (m represents the size of the smallest subunit: m = 2 when the smallest subunit comprises 2 x 2 processors), and j (step S22). That is, a decision is made as to whether or not the destination processor is in a smallest subunit at the left of the switch.

When $i - m < dest < j$ (YES, i.e., the destination processor is in a smallest subunit at the left of the switch), the switch is controlled so as to link terminal 1 and terminal 2 together (step S23).

When not $i - m < dest < j$ (NO), a comparison is then made between dest and $j + m$ (step S24). That is, a decision is made as to whether or not the destination processor is in a smallest subunit at the right of the switch. When $dest < j + m$ (YES, i.e., the destination processor is in a smallest subunit at the right of the switch), the terminal via which the message is routed is linked to terminal 4 (step S25). When $dest \geq j + m$ (NO), on the other hand, that terminal is linked to terminal 3 (step S26).

When the message is routed via terminal 3 or terminal 4, on the other hand, a comparison is made between dest and i (step S27). That is, a decision is made as to whether or not the destination processor is to the right of the switch. When $i < dest$ (YES, i.e., the destination processor is to the right of the switch), terminal 3 and terminal 4 are linked together (step S28). When $i \geq dest$ (NO, i.e., the destination processor is to the left of the switch), on the other hand, a comparison is then made between dest and $j - n$ (step S29). This makes a decision of whether or not the destination processor is in the subunit immediately to the left of the switch. If dest > j - m (YES, i.e., the destination processor is in the subunit immediately to the left of the switch), then terminal 2 is made the output terminal (step S30). If dest ≤ j - m (NO), then terminal 1 is made the output terminal (step S31).

The above-described header analyzing method permits a message to be sent to any destination processor over the shortest route.

The use of such message transmission-based switching as described above permits high-speed communications among processor elements in a parallel computer. This could also be utilized to establish synchronization of all or part of processors in a parallel computer.

In the event of the occurrence of a failure in part of processor elements in a parallel computer, it would also be possible to change over switches so as to isolate a subtorus containing a failed processor element, continue processing by the use of other subtori, and repair or replace the failed processor element in the isolated subtorus.

C L A I M S:

1. A reconfigurable n-dimensional torus network system for use in a parallel computer comprising:

a plurality of processor element subsets, each comprising at least one processor element;

a switch means, provided between adjacent processor element subsets; and

an interleave connection means for connecting a first processor element subset to a second processor element subset by shipping a third processor element subset through said switch means.

2. The reconfigurable torus network according to Claim 1 wherein:

said switch means comprised a THROUGH state in which said first processor element subset is connected to said second processor element subset by said interleaving connection means and a TURN state in which said first processor element subset is provided with a return path.

3. The reconfigurable torus network according to Claim 1 wherein said switch means comprises first and second terminals connected to the adjacent processor

element subsets and third and fourth terminals connected to adjacent switch means.

4. The reconfigurable torus network according to Claim 1 wherein said interleaving connection means comprises or communicative path wire.

5. The reconfigurable torus network system according to Claim 1 further comprising:

a switch control means for changing over said switch means to split said torus into subtori or to integrate split subtori into said torus network thereby reconfigurable said torus network.

6. The reconfigurable torus network system according to Claim 5, in which said switch control means changes over said switch means before a user uses said parallel computer and fixes the states of said switches while said user is using said parallel computer, thereby splitting/integrating said torus network statically.

7. The reconfigurable torus network system according to Claim 5, said switch control means changes over said switch means as requested, thereby

splitting/integrating said torus network dynamically.

8. The reconfigurable torus network system according to Claim 5, in which said switch control means changes over said switch means so as to connect a communications path between said switch means and a communications path thereby shortening communications distance between processor elements which are distant from each other.

9. The reconfigurable torus network system according to Claim 7, in which said switch control means changes over said switches in response to a message that is created by said user and then sent out to said network.

10. The reconfigurable torus network system according to Claim 9, in which the number of processor elements included in said torus network reconfigured by split/integration in said message is specified, and said switching means changes over said switches in accordance with the contents of said message.

11. The reconfigurable torus network system according to Claim 9, in which the identification number of a

processor element is specified upon communication, and said switching means changes over said switches in accordance with the contents of said message.

12. The reconfigurable torus network system according to Claim 8, further comprising means for permitting high-speed communications among processor elements to thereby perform high-speed global calculations in all processor elements included in said parallel computer.

13. The reconfigurable torus network system according to Claim 8, further comprising means for permitting high-speed communications among processor elements to thereby perform high-speed broadcast processing on all processor elements included in said parallel computer.

14. The reconfigurable torus network system according to Claim 8, in which high-speed communications are performed among processor elements and said switches are changed over dynamically to thereby perform high-speed broadcast processing on part of processor elements included in said parallel computer.

15. The reconfigurable torus network system according to Claim 8, further comprising means for performing

high-speed communications among processor elements and providing synchronization of all processor elements included in said parallel computer at high speed.

16. The reconfigurable torus network system according to Claim 8, in which high-speed communications are performed among processor elements and said switches are changed over dynamically to thereby provide synchronization of part of processor elements included in said parallel computer at high speed.

17. The reconfigurable torus network system according to Claim 1, in which in the event of a processor failure, said network is split to isolate a failed processor element for subsequent replacement.

18. A reconfigurable n-dimensional torus network system substantially as hereinbefore described with reference to Figures 2 to 11, excluding Figure 3A, of the accompanying drawings.

Patents Act 1977 **E[illegible]niner's report to the Comptroller under Section 17 (The Search report)** -36-	**Application number** **GB 9405556.3**
Relevant Technical Fields (i) UK Cl (Ed.M) G4A (AFGL, AMP) (ii) Int Cl (Ed.5) GO6F 15/16	**Search Examiner** **S J PROBERT**
	Date of completion of Search **3 JUNE 1994**
Databases (see below) (i) UK Patent Office collections of GB, EP, WO and US patent specifications. (ii) ONLINE: WPI	**Documents considered relevant following a search in respect of Claims :-** **1-18**

Categories of documents

X:	Document indicating lack of novelty or of inventive step.	P:	Document published on or after the declared priority date but before the filing date of the present application.
Y:	Document indicating lack of inventive step if combined with one or more other documents of the same category.	E:	Patent document published on or after, but with priority date earlier than, the filing date of the present application.
A:	Document indicating technological background and/or state of the art.	&:	Member of the same patent family; corresponding document.

Category	Identity of document and relevant passages	Relevant to claim(s)
X	EP 0588104 A2 (IBM) see abstract	1 at least
A	EP 0544532 A2 (FUJITSU LTD) see whole document	1-18

Databases: The UK Patent Office database comprises classified collections of GB, EP, WO and US patent specifications as outlined periodically in the Official Journal (Patents). The on-line databases considered for search are also listed periodically in the Official Journal (Patents).